Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

**Отчёт по программе «Практическая работа №4. Индивидуальные задания»**

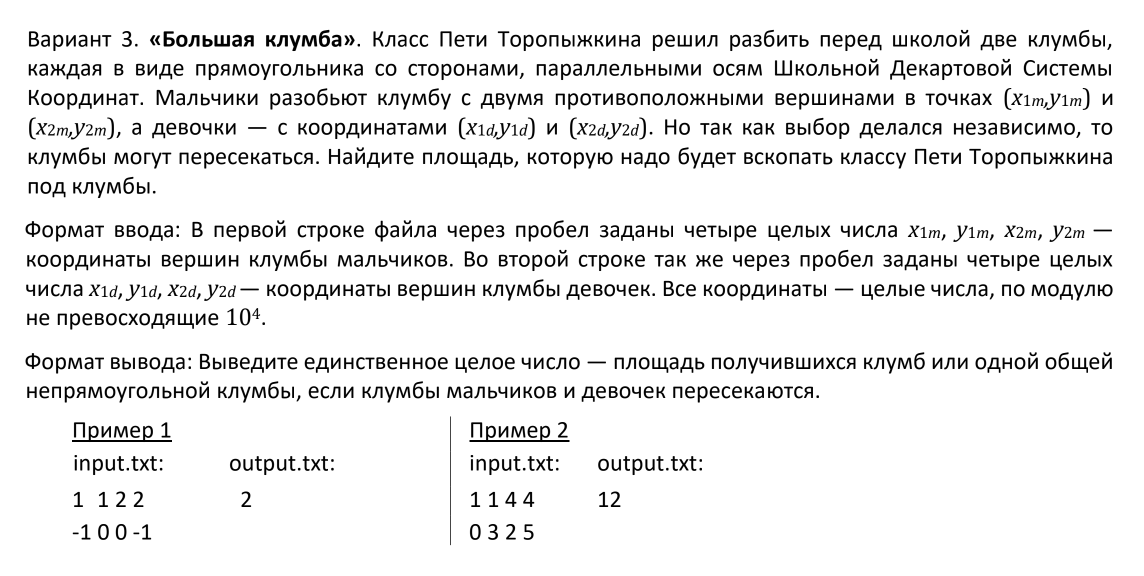
Выполнила: Мезенцева Полина Сергеевна

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2024

**Задание 4.2 (Индивидуальные задания). Вариант 3:**



**Входные данные**

str – string[] - массив строк, считываемый с файла input.txt

**Выходные данные**

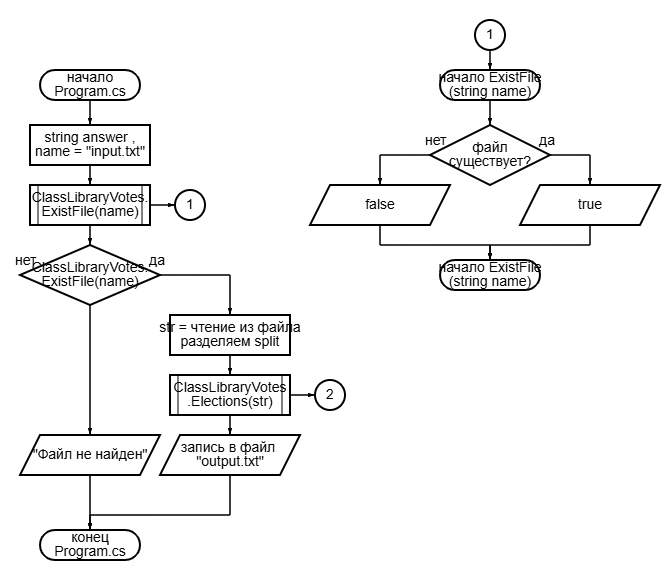
totalArea - int – переменная, в которой хранится итоговое значение площади, выводится в файл output.txt

**Описание разработанных функций и методов:**

Методы класса ClassLibraryFlowerbad:

|  |  |
| --- | --- |
| bool Check(string[] str) | Метод проверки полученного массива str на корректность; Присваивание свойствам значения из массива. |
| int RectangleArea(string who\_lower) | В зависимости от того, кто приходит в метод, используются нужные значения нужных свойств. |
| int IntersectionArea() | Функция, которая определяет рамки общей площади клумб. |
| Bool ExistFile(string name) | Метод определения существования файла. |

**Блок-схема**

****

**Листинг программы**

Program.cs

class Program

{

static void Main(string[] args)

{

string answer , name = "input.txt";

string[] str;

if (ClassLibraryVotes.ExistFile(name))

{

using (var reader = new StreamReader(name))

{

str = reader.ReadToEnd().Split(new[] { '\n', '\r', ' ' }, StringSplitOptions.RemoveEmptyEntries);

answer = ClassLibraryVotes.Elections(str);

File.WriteAllText("output.txt", answer);

}

}

else Console.WriteLine("Файл не найден");

}

}

Class1.cs

public class ClassLibraryFlowerbad

{

static int X1m { get; set; }

static int X2m { get; set; }

static int Y1m { get; set; }

static int Y2m { get; set; }

static int X1d { get; set; }

static int X2d { get; set; }

static int Y1d { get; set; }

static int Y2d { get; set; }

public static bool Check(string[] str)

{

if (str.Length == 8)

{

if (int.TryParse(str[0].ToString(), out int x1m) &&

int.TryParse(str[1].ToString(), out int y1m) &&

int.TryParse(str[2].ToString(), out int x2m) &&

int.TryParse(str[3].ToString(), out int y2m) &&

int.TryParse(str[4].ToString(), out int x1d) &&

int.TryParse(str[5].ToString(), out int y1d) &&

int.TryParse(str[6].ToString(), out int x2d) &&

int.TryParse(str[7].ToString(), out int y2d))

{

X1m = x1m;

Y1m = y1m;

X2m = x2m;

Y2m = y2m;

X1d = x1d;

Y1d = y1d;

X2d = x2d;

Y2d = y2d;

return true;

}

else return false;

}

else return false;

}

public static int RectangleArea(string who\_lower)

{

if (who\_lower == "boys") return Math.Abs(X2m - X1m) \* Math.Abs(Y2m - Y1m);

else if (who\_lower == "girls") return Math.Abs(X2d - X1d) \* Math.Abs(Y2d - Y1d);

else return 0;

}

public static int IntersectionArea()

{

int xLeft = Math.Max(X1m, X1d);

int xRight = Math.Min(X2m, X2d);

int yBottom = Math.Max(Y1m, Y1d);

int yTop = Math.Min(Y2m, Y2d);

if (xLeft < xRight && yBottom < yTop)

{

return (xRight - xLeft) \* (yTop - yBottom);

}

return 0;

}

public static bool ExistFile(string name)

{

if (File.Exists(name)) return true;

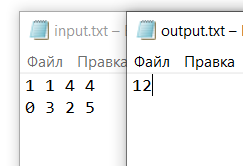
return false;

}

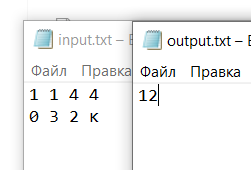
}

**Тестовые ситуации**

1) В файле ввода правильная информация, выводится верная информация



2) В файле ввода ошибка(ничего не изменилось)



**Модульные тесты:**

**Листинг тестов:**

using ClassLibrary1;

namespace TestProject2

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethodCheckError1()

{

string[] str = { "1", "1", "4", "4", "0", "3", "5", "2", "9" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheckError2()

{

string[] str = { "1", "1", "4", "4", "0", "3", "5", "2", "девять" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

//

[TestMethod]

public void TestMethodCheck1()

{

string[] str = { "w", "1", "4", "4", "0", "3", "5", "2"};

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck2()

{

string[] str = { "1", "w", "4", "4", "0", "3", "5", "2"};

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck3()

{

string[] str = { "1", "1", "w", "4", "0", "3", "5", "2" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck4()

{

string[] str = { "1", "1", "4", "w", "0", "3", "5", "2" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck5()

{

string[] str = { "1", "1", "4", "4", "w", "3", "5", "2" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck6()

{

string[] str = { "1", "1", "4", "4", "0", "w", "5", "2" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck7()

{

string[] str = { "1", "1", "4", "4", "0", "3", "w", "2" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodCheck8()

{

string[] str = { "1", "1", "4", "4", "0", "3", "5", "w" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

//

[TestMethod]

public void TestMethodCheck()

{

string[] str = { "1", "1", "t", "4", "0", "3", "5", "t" };

Assert.AreEqual(false, ClassLibraryFlowerbad.Check(str));

}

[TestMethod]

public void TestMethodRectangleAreaError()

{

string[] str = { "1", "t", "4", "4", "0", "3", "5", "t" };

ClassLibraryFlowerbad.Check(str);

Assert.AreEqual(0, ClassLibraryFlowerbad.RectangleArea("кто-то"));

}

[TestMethod]

public void TestMethodRectangleAreaBoys()

{

string[] str = { "1", "1", "4", "4", "0", "3", "5", "2" };

ClassLibraryFlowerbad.Check(str);

Assert.AreEqual(9, ClassLibraryFlowerbad.RectangleArea("boys"));

}

[TestMethod]

public void TestMethodRectangleAreaGirls()

{

string[] str = { "1", "1", "4", "4", "0", "3", "5", "2" };

ClassLibraryFlowerbad.Check(str);

Assert.AreEqual(5, ClassLibraryFlowerbad.RectangleArea("girls"));

}

[TestMethod]

public void TestMethodIntersectionAreaError1()

{

string[] str = { "3", "1", "3", "2", "0", "3", "5", "2" };

ClassLibraryFlowerbad.Check(str);

Assert.AreEqual(0, ClassLibraryFlowerbad.IntersectionArea());

}

[TestMethod]

public void TestMethodIntersectionAreaError2()

{

string[] str = { "1", "1", "4", "4", "0", "3", "2", "2" };

ClassLibraryFlowerbad.Check(str);

Assert.AreEqual(0, ClassLibraryFlowerbad.IntersectionArea());

}

[TestMethod]

public void TestMethodIntersectionArea()

{

string[] str = { "1", "1", "4", "4", "0", "3", "2", "5" };

ClassLibraryFlowerbad.Check(str);

Assert.AreEqual(1, ClassLibraryFlowerbad.IntersectionArea());

}

[TestMethod]

public void TestMethodAreaTotal()

{

string[] str = { "1", "1", "4", "4", "0", "3", "5", "2" };

ClassLibraryFlowerbad.Check(str);

int areaBoys = ClassLibraryFlowerbad.RectangleArea("boys");

int areaGirls = ClassLibraryFlowerbad.RectangleArea("girls");

int totalArea = areaBoys + areaGirls - ClassLibraryFlowerbad.IntersectionArea();

Assert.AreEqual(14, totalArea);

}

[TestMethod]

public void TestMethodExistFileFalse()

{

Assert.AreEqual(false, ClassLibraryFlowerbad.ExistFile("output.txt"));

}

[TestMethod]

public void TestMethodExistFileTrue()

{

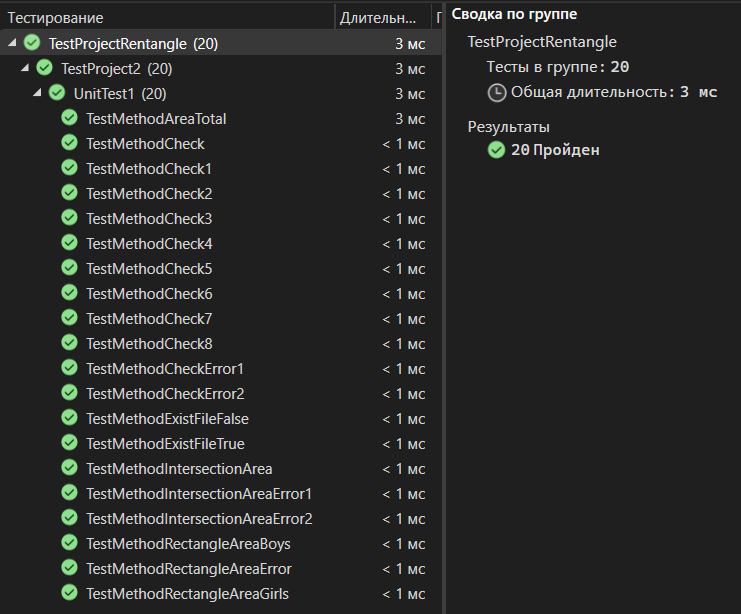
Assert.AreEqual(true, ClassLibraryFlowerbad.ExistFile("input.txt"));

}

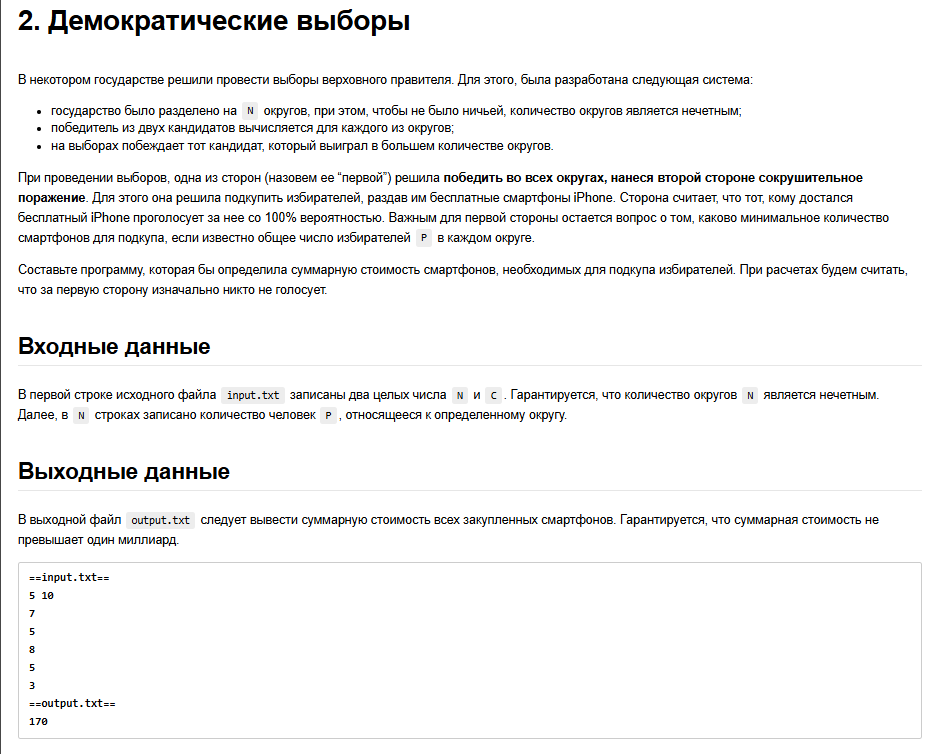
}

}

**Результаты прохождения тестов:**



**Задание 4.2 (text.html). Вариант 2**

**:**

**Входные данные**

str – string[] - массив строк, считываемый с файла input.txt

**Выходные данные**

answer - string – переменная, в которой хранится итоговую сумму для подкупа, выводится в файл output.txt

**Описание разработанных функций и методов:**

Методы класса ClassLibraryFlowerbad:

|  |  |
| --- | --- |
| String Elections(string[] str) | Метод проверки полученного массива str на корректность; Расчет общей суммы взятки. |
| Bool ExistFile(string name) | Метод определения существования файла. |

**Блок-схема**

**Листинг программы**

Class1.cs

public class ClassLibraryVotes

{

public static string Elections(string[] str)

{

if (str.Length > 0)

{

if (str[0] != "0")

{

var all = str.All(t => int.TryParse(t, out \_));

if (all == true)

{

var all2 = str.All(t => int.Parse(t) > 0);

if (all2 == true)

{

if (int.Parse(str[0]) % 2 == 1)

{

if (str.Length - 2 == int.Parse(str[0]))

{

int N = int.Parse(str[0]);

int C = int.Parse(str[1]);

int allVotes = 0;

for (int i = 2; i <= N + 1; i++)

{

int P = int.Parse(str[i]);

int neededVotes = P / 2 + 1;

allVotes += neededVotes;

}

int totalCost = allVotes \* C;

if (totalCost <= 10000000) { return totalCost.ToString(); }

else return "error 6";

}

else return "error 4";

}

else return "error 3";

}

else return "error 5";

}

else return "error 2";

}

else return "error 1\_2";

}

else return "error 1";

}

public static bool ExistFile(string name)

{

if (File.Exists(name)) return true;

return false;

}

}

Program.cs

class Program

{

static void Main(string[] args)

{

string answer , name = "input.txt";

string[] str;

if (ClassLibraryVotes.ExistFile(name))

{

using (var reader = new StreamReader(name))

{

str = reader.ReadToEnd().Split(new[] { '\n', '\r', ' ' }, StringSplitOptions.RemoveEmptyEntries);

answer = ClassLibraryVotes.Elections(str);

File.WriteAllText("output.txt", answer);

}

}

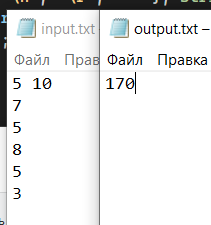
else Console.WriteLine("Файл не найден");

}

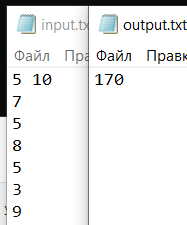
}

**Тестовые ситуации**

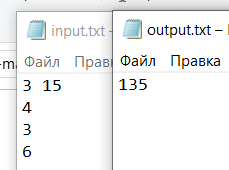
1) В файле ввода правильная информация, выводится верная информация



2) В файле ввода ошибка(ничего не изменилось)



3) Опять правильный ввод



**Модульные тесты:**

**Листинг тестов:**

using ClassLibrary1;

namespace TestProject1

{

public class Tests

{

[Test]

public void TestError1\_1()//проверка заполненности файла (не заполнен)

{

string[] input = { };

Assert.AreEqual("error 1", ClassLibraryVotes.Elections(input));

}

//

[Test]

public void TestError1\_2()//проверка заполненности файла (количество равно 0)

{

string[] input = { "0", "10" };

Assert.AreEqual("error 1\_2", ClassLibraryVotes.Elections(input));

}

//

[Test]

public void TestError2\_2()//проверка все ли элементы int (не все)

{

string[] input = { "5", "10", "7", "w", "8", "5", "3" };

Assert.AreEqual("error 2", ClassLibraryVotes.Elections(input));

}

//

[Test]

public void TestError3\_2()//проверка на нечетность(четное количество)

{

string[] input = { "6", "10", "7", "5", "8", "5", "3", "6" };

Assert.AreEqual("error 3", ClassLibraryVotes.Elections(input));

}

//

[Test]

public void TestError4\_2()//проверка на совпадение количества и числа фактических республик(не совпадает)

{

string[] input = { "5", "10", "7", "5", "8", "5", "3", "6" };

Assert.AreEqual("error 4", ClassLibraryVotes.Elections(input));

}

//

[Test]

public void Test\_NegativeVotes()

{

string[] input = { "-3", "10", "5", "5", "5" }; //отрицательные республики недопустимы

string result = ClassLibraryVotes.Elections(input);

Assert.AreEqual("error 5", result);

}

[Test]

public void Test\_ValidInput\_OneVoteNegativeCost() //отрицательная стоимость недопустима

{

string[] input = { "1", "-10", "5" };

string result = ClassLibraryVotes.Elections(input);

Assert.AreEqual("error 5", result);

}

[Test]

public void Test\_ValidInput\_ThreeVotesWithNegative()// Отрицательные голоса недопустимы

{

string[] input = { "3", "5", "-1", "1", "1" };

string result = ClassLibraryVotes.Elections(input);

Assert.AreEqual("error 5", result);

}

[Test]

public void Test\_ValidInput\_ThreeVotesWithZero()// Все голоса равны нулю

{

string[] input = { "3", "5", "0", "0", "0" };

string result = ClassLibraryVotes.Elections(input);

Assert.AreEqual("error 5", result);

}

[Test]

public void Test\_ValidInput\_OneVoteWithZero()// Стоимость равна нулю

{

string[] input = { "1", "10", "0" };

string result = ClassLibraryVotes.Elections(input);

Assert.AreEqual("error 5", result);

}

[Test]

public void Test\_ValidInput\_LargeNumbers() //итоговая сумма превышает 1 миллиард

{

string[] input = { "3", "10000", "1000", "1000", "1000" };

Assert.AreEqual("error 6", ClassLibraryVotes.Elections(input));

}

[Test]

public void Test\_ValidInput\_Answer()//тест на пример из задания

{

string[] input = { "5", "10", "7", "5", "8", "5", "3" };

Assert.AreEqual("170", ClassLibraryVotes.Elections(input));

}

[Test]

public void Test\_ValidInput\_SingleVote()

{

string[] input = { "1", "10", "5" }; //тест с одним голосом

Assert.AreEqual("30", ClassLibraryVotes.Elections(input));

}

[Test]

public void Test\_ValidInput\_ThreeVotes()

{

string[] input = { "3", "5", "1", "1", "1" };

Assert.AreEqual("15", ClassLibraryVotes.Elections(input));

}

[Test]

public void Test\_ValidInput\_ThreeVotesWithLargeNumbers()

{

string[] input = { "3", "100", "1000", "1000", "1000" };

Assert.AreEqual("150300", ClassLibraryVotes.Elections(input)); // (1000/2 + 1) \* 100 \* 3 = 150000

}

//

[Test]

public void TestExistFileFalse()//файл не существует

{

Assert.AreEqual(false, ClassLibraryFlowerbad.ExistFile("output.txt"));

}

[Test]

public void TestExistFileTrue() //файл существует

{

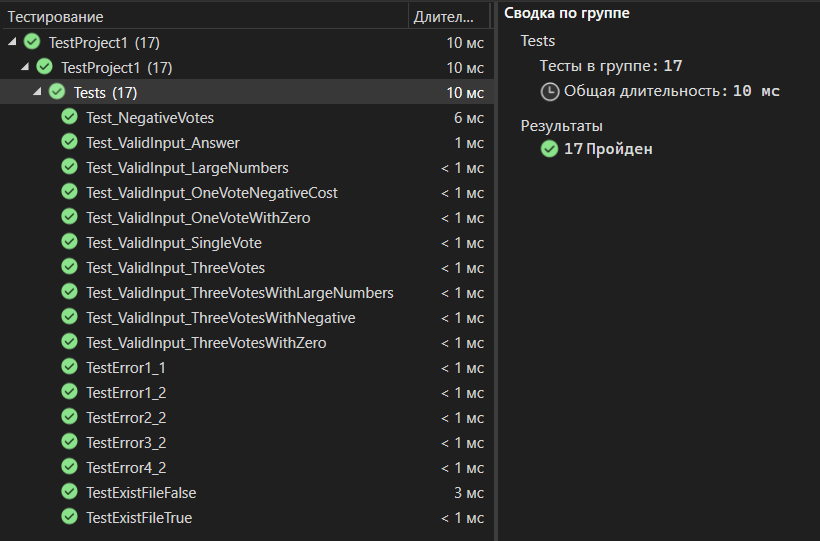
Assert.AreEqual(true, ClassLibraryFlowerbad.ExistFile("input.txt"));

}

}

}

**Результаты прохождения тестов:**



**Используемые библиотеки:**

|  |  |
| --- | --- |
| System | Содержит основные классы и базовые классы, определяющие часто используемые значения и ссылочные типы данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки. |
| Math | Предоставляет константы и статические методы для тригонометрических, логарифмических и других распространенных математических функций. |

**Используемые инструменты:** Visual Studio TestUnit C#, LibraryClasses C#, Console C#.